Institut national des sciences appliquées de Rouen

Rapport Projet BDD

Gestion d'une boutique en ligne

Auteurs :
Bahaeddine Hilal
bahaeddine.hilal@insa-rouen.fr
Thomas Boeno
thomas.boeno@insa-rouen.fr
Redouane Bouazza
redouane.bouazza@insa-rouen.fr

Enseignant :
Nathalie Chaignaud
natalie.chaignaud@insa-rouen.fr

Mai 2022

Table des matières

1. Modélisation

1.1 Introduction

Nous proposons de modéliser une boutique en ligne de type Amazon. Depuis la crise du covid, les boutiques en ligne connaissent un succès fulgurant. En effet, à la suite du confinement général de la population et de la fermeture des boutiques traditionnelles, les clients ont été contraints de repenser leurs modes de consommation. Naturellement, ils se sont alors tournés vers les boutiques en ligne qui proposent un service fiable et rapide de livraison à domicile. Manipulant une quantité importante de données, ces sites ne peuvent se permettre de négliger leur traitement. C'est ce qui nous a motivés à penser un modèle simplifié pour la gestion de ces données.

Dans notre modèle, la boutique propose un ensemble de produits regroupés dans différentes catégories. Les clients doivent avoir un compte client pour pouvoir commander un ou plusieurs produits. Afin de pouvoir passer une commande, le client devra aussi avoir renseigné au préalable un moyen de paiement, ici une carte bancaire, et une adresse pour la livraison. Un produit aura aussi un fournisseur associé qui sera chargé de livrer le client. Nous pourrons de plus consulter les commandes qui auront déjà été effectuées.

Exemple requête :

- Liste des ordinateurs portables en dessous de 1000e.
- Liste des dates avec le plus de ventes.
- Liste des objets dans une catégorie triée par prix.
- Liste des produits les plus vendus avec leurs fournisseurs.
- Liste des fournisseurs n'effectuant aucune vente.
- Liste des produits achetés par un client donné pour lui en proposer d'autres.

1.2 Graphe

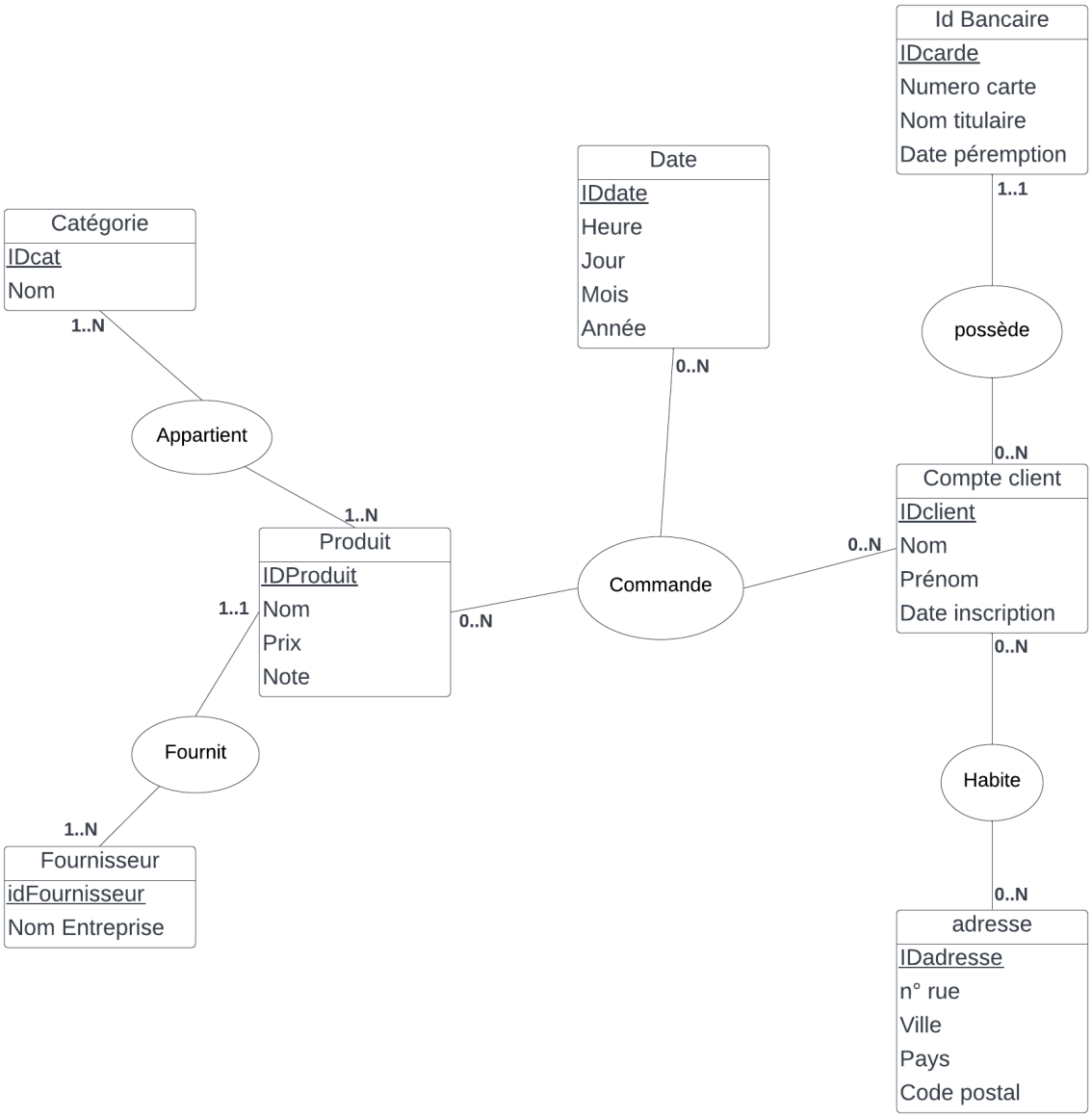

FIGURE 1.1 – Modèle Entité Association

1.3 Les Entités

Compte Client : Le compte client donne des informations sur le compte d'un client. Chaque compte client est identifié par une clef IdClient. On y retrouve aussi le nom, le prénom et la date d'inscription d'un client.

Date : Pour définir une date, on a donc besoin des attributs suivants : heure, jour, mois, année. On ajoute aussi une clef IdDate pour pouvoir identifier une date plus facilement.

Catégorie : Elle regroupe les catégories de produits (ex : High-tech, Vêtements, Bricolage ...). Ces catégories pourront ensuite être utilisées pour rechercher un type précis de produit. On y trouve la clef IdCat et le nom de la catégorie.

Identité Bancaire : Informations bancaires associées à une carte. Les attributs présents au sein de cette entité sont donc le numéro de carte, le titulaire, la date de péremption et une clef primaire : IdCarte.

Produit : Il donne des informations sur les produits. On y retrouve la clef IdProduit, le nom du produit, son prix et la moyenne des notes que ce produit a reçue.

Fournisseur : Afin de simplifier le modèle, on considère ici que le fournisseur est aussi le constructeur du produit fourni. On dispose alors d'un nom d'entreprise ainsi qu'une clef IdFournisseur.

Adresse : Elle décrit l'adresse d'un client. Nous identifions une adresse avec la clef IdAdresse, et les autres attributs sont le n° de la rue, la ville et le pays.

1.4 Les Associations

Commande : Une commande relie un produit, un compte client et une date précise. Pour une date donnée, aucun à plusieurs produits ont pu être commandés, et ces commandes ont pu être réalisées par un ou plusieurs clients. De même un compte client peut commander sur une à plusieurs dates (voire aucune) et peut commander 1 à plusieurs produits. Pour finir un produit peut être commandé par aucun,un seul ou plusieurs clients. Et ce produit peut être commandé à plusieurs dates ou aucune. (à compléter)

Appartient : Un produit appartient à une ou plusieurs catégories générales (High-tech, Vêtements, Bricolage ...) et une catégorie peut contenir un ou plusieurs produits. Exemple : Une chaise Gaming peut appartenir en même temps à la catégorie High-Tech et Ameublement.

Fournit : Un fournisseur est responsable de fournir un ou plusieurs produits (Intel par exemple fournit des processeurs, des ordinateurs portables et des drones). Un produit ne peut avoir qu'un et un seul fournisseur que nous considérons ici aussi comme constructeur. En effet, pour deux produits donnés même s'ils remplissent le même rôle (chargeur par exemple) les fiches techniques seront différentes et ainsi les produits auront des identifiants différents.

Possède : Un compte client peut avoir une ou plusieurs cartes (exemple de compte pour un membre avec plusieurs identités bancaires) ou même n'en avoir aucune s'il n'a pas encore effectué une commande. Toutefois, une carte ne peut être associée qu'à un et un seul compte bancaire, pour protéger l'utilisateur et le notifier si sa carte est utilisée par un autre compte. De plus, on peut supposer que les couples/familles qui n'ont qu'une carte commune, ont un seul compte.

Habite : Un compte client peut posséder plusieurs adresses dans le cas d'un étudiant par exemple qui peut être livrée soit chez lui, soit chez ses parents. Ou bien il peut ne pas avoir d'adresse s'il n'a pas encore effectué de commande. Une adresse n'a pas besoin d'être stockée si aucun client n'y habite, et une même adresse peut appartenir é plusieurs clients (comme par exemple le cas d'un appartement en collocation).

1.5 Conception du modèle relationnel :

Dans cette partie, nous passons d'un schéma Entité/Association au schéma relationnel.

Pour chaque entité nous créons une relation de même nom avec les mêmes propriétés et l'identifiant constitue la clé.

- Compte Client (IdClient, Nom, Prénom, Date d'inscription)
- Date (IdDate, Heure, Jour, Mois, Année)
- Catégorie (IdCatég, NomCatég)
- Identité Bancaire (IdCarte, Numéro, DatePéremption)
- Produit (IdProduit, Nom, Prix, MoyenneNotes)
- Fournisseur (IdFournisseur, NomEntreprise)
- Adresse(IdAdresse, numéro, rue, ville, pays)

Pour l'association ternaire, on rajoute un identifiant qui caractérise de manière unique l'occurrence pour éviter d'utiliser les trois clés primaires.

- Commande(nCommande, DateCommande, IdProduit, IdClient)
- Appartenir(IdProduit, IdCatég,)
- Fournir(IdFournisseur,IdProduit, Quantité)
- Posséder(IdClient, IdCarte)
- Habiter(IdClient, IdAdresse)

1.6 Dénormalisation du modèle logique

On tente de supprimer les entités portant peu d'attributs :

1. Il est inutile de garder une entité date, il suffit de stocker la date dans la facture.

Ainsi, on aura les changements suivants sur les entités :

— ~~Date(IdDate,Heure,Jour,Mois,Année)~~

— Produit (IdProduit, Nom, Prix, MoyenneNotes,**NomEntreprise**, **NomCatég**)

— Commande(néCommande, DateCommande, IdProduit, IdClient,)

On introduit la redondance :

Notons que si nous souhaitons par exemple déterminer quel fournisseur livre à une adresse, nous devons parcourir les tables Compte Client et Produit, donc on propose de rajouter l'attribut adresse dans l'entité Commande.

Et donc on aura :

— Commande (nCommande, DateCommande, IdProduit, IdClient, **AdresseLivraison**)

2. Implantation

2.1 Creation de la base de donnée

2.1.1 Création des tables

- Create table Produit(IDProduit integer,Nom varchar(30) not null,Prix float, Note float, primary key(IDProduit))
- Create table Fournisseur(IDFournisseur integer, nomEntrep varchar(30) not null, primary key (IDFournisseur))
- Create table Fournit(IDPro integer, IDFourn integer, foreign key(IDPro) references Produit(IDProduit), foreign key (IDFourn) references Fournisseur(IDFournisseur))
- Create table Categorie(IDCategorie integer, NomC varchar(30) not null, primary key (IDCategorie))
- Create table Appartient(IDprod integer, IDcat integer, foreign key(IDprod) references Profuit(IDProduit), foreign key(IDcat) references Categorie(IDCategorie))
- Create table IDBancaire(IDCarte integer, Nom varchar(30) not null, NumeroCarte integer not null, DateExpiration DATE, primary key(IDCarte))
- Create table CompteClient(IDClient integer, Nom varchar(30) not null, primary key(IDClient))
- Create table Possede(IDClient integer, IDbc, foreign key(IDClient) references CompteClient(IDClient), foreign key(IDbc) references IDBancaire(IDCarte))
- Create table Adresse(IDAdresse integer, Numero integer, Rue varchar(30) not null, Ville varchar(30) not null, Pays varchar(30) not null, primary key(IDAdresse))
- Create table Habite(IDClient integer, IDAdresse integer, foreign key(IDClient) references CompteClient(IDClient), foreign key(IDAdresse) references Adresse(IDAdresse))
- Create table Commande(nCommande integer, DateCommande DATE, IDProduit integer, IDClient integer, AdresseLivraison integer, primary key (nCommande), foreign key(IdProduit) references Produit(IDProduit), foreign key(IdClient) references CompteClient(IDClient), foreign key(AdresseLivraison) references Adresse(IDAdresse))

2.1.2 Ajout des données

Voici quelques que exemples de commande pour ajouter des données. Étant donné le nombre de donnée, nous utiliserons l'outil *sqlitebrowser* qui permet l'importaion de données depuis un fichier *.csv* que l'on peut remplir plus facilement à l'aide d'un tableur.

- Produit :
  insert into Produit(IDProduit,Nom,Prix,Note) values (10201,"Acer Predator",800,4.5) ;
- Fournisseur :
  insert into Fournisseur(IDFournisseur,nomEntrep) values (10301,"Decathlon") ;
- Fournit :
  insert into Fournit(IDPro,IDFourn) values (10201,10312) ;
- Categorie :
  insert into Categorie(IDCategorie,NomC) values (10101,"Ameublement") ;
- Appartient :
  insert into Appartient(IDprod,IDcat) values (10201,10103) ;
- IDBancaire :
  insert into IDBancaire(IDCarte,Nom,NumeroCarte,DateExpiration) values (10501,"Kevin",90872,09/24) ;
- CompteClient :
  insert into CompteClient(IDClient,Nom) values (10601,"Kevin") ;
- Possede :
  insert into Possede(IDClient,IDbc) values (10601,10501) ;
- Adresse :
  insert into Adresse(IDAdresse,Numero,Rue,Ville,Pays) values (10701,15,"Simone signoret","Rouen","France") ;
- Habite :
  insert into Habite(IDClient,IDAdresse) values (10601,10718) ;
- Commande :
  insert into Commande(nCommande,DateCommande,IdProduit,IdClient,AdresseLivraison) values (14000,01/01/2018,10216,10601,10718) ;

2.2 Algèbre relationnelle

Liste des ordinateurs en dessous de 1000 euros :

$$\underset{Nom,NomC,prix}{\pi}\left(\underset{\substack{NomC=PC\ portable\\ prix<1000}}{\sigma}\left(Produit \underset{IDProduit=IDprod}{\bowtie} Appartient\right) \underset{IDcat=IDCategorie}{\bowtie} Categorie\right)$$

Requête utilisant l'opérateur IN : Liste des commandes en Europe :

$$C = \left(\left(CompteClient \underset{IDclient=IDclient}{\bowtie} Commande\right) \underset{IDclient=IDclient}{\bowtie} Habite\right) \underset{IDadr=IDadr}{\bowtie} Adresse$$
$$\underset{C.Nom,ncommande,Pays,...}{\pi}\left(\underset{pays="France"\vee pays="Italie"\vee pays=...}{\sigma}(C)\right)$$

Requête utilisant l'opérateur NOT IN : Liste des commandes hors Afrique

$$C = \left(\left(CompteClient \underset{IDclient=IDclient}{\bowtie} Commande\right) \underset{IDclient=IDclient}{\bowtie} Habite\right) \underset{IDadr=IDadr}{\bowtie} Adresse$$
$$\underset{C.Nom,ncommande,Pays,...}{\pi}\left(\underset{pays!="Maroc"\wedge pays!="Senegal"\wedge pays!=...}{\sigma}(C)\right)$$

Requête utilisant l'opérateur COUNT : Trouver les clients ayant effectués le plus de commandes (pour des offres promotionnelles)

$$\underset{nb\_commandes\downarrow}{\tau}\left(\underset{COUNT(IDClient)\rightarrow nb\_commandes,CompteClient.nom}{\pi}\left(\underset{idclient,COUNT(idclient)}{\sigma}(C)\right)\right)$$
$$C = \left(\left(Produit \underset{IPDroduit}{\bowtie} commande\right) \underset{IDClient}{\bowtie} compteclient\right)$$

Requête utilisant l'opérateur GROUP BY : Les fournisseurs ayant fait le meilleure chiffre d'affaire

$$\underset{\substack{Fournisseur.nomEntrep\\ COUNT(idproduit)\rightarrow nb\_produit\\ SUM(prix)\rightarrow chiffre\_affaire}}{\pi}\left(\underset{\substack{IDFournisseur,SUM(prix)\\ COUNT(IDProduit)}}{\gamma}\left(\underset{Fournisseur.idfournisseur=fournit.IDfourn}{\underset{Fournit.IDpro=Produit.IDProduit}{\sigma}}(C)\right)\right)$$
$$C = Fournit \times Fournisseur \times \left(Produit \underset{IDProduit}{\bowtie} Commande\right)$$

Requête utilisant l'opérateur EXISTS : Les utilisateurs ayants effectué au moins une commande

$$\underset{client,nom}{\pi}\left(CompteClient \underset{IDClient}{\bowtie} Commande\right)$$

Requête utilisant l'opérateur NOT EXISTS : Les client n'ayant effectué aucune commande

$$Client - \underset{idclient,nom}{\pi}\left(CompteClient \underset{IDClient}{\bowtie} Commande\right)$$

Requêtes IMBRIQUES : Client ayant dépensé plus de 1000 euros

$$\underset{\substack{CompteClient.nom \\ SUM(prix) \to depense}}{\pi} \left(\underset{\substack{SUM(prix) \\ IDClient}}{\gamma} \left(\underset{depense>=1000}{\sigma} \left(CompteClient \underset{IDClient}{\bowtie} Commande \underset{IDProduit}{\bowtie} Produit \right) \right) \right)$$

Sous Requêtes corrélées : rechercher tous les fournisseur dont les ventes sont supérieur à 10 % des ventes totales.

$$\underset{0.1 \times SCA < CA}{\sigma} (A \times B)$$

$$A = \underset{\substack{SUM(prix) \to CA \\ nomEntrep}}{\pi} \left(\underset{\substack{Sum(prix) \\ IDFournisseur}}{\gamma} \left(Commande \underset{IDProduit}{\bowtie} Produit \underset{Idproduit}{\bowtie} Fournit \underset{IDFournisseur}{\bowtie} Fournisseur \right) \right)$$

$$B = \underset{SUM(prix) \to SCA}{\pi} \left(Commande \underset{IDProduit}{\bowtie} Produit \right)$$

Requête utilisant l'opérateur DIVISION : Liste des clients ayant commandé toutes les catégories

$$Client \div Categorie$$

Liste des fournisseurs fournissant toutes les catégories

$$Fournisseur \div Categorie$$

Jointure avec deux fois la même table : Liste des produits commandés par au moins deux utilisateurs

$$\underset{NomClient}{\pi} \left(Commande \underset{\substack{IDProduit = IDProduit \\ IDClient \neq IDClient}}{\bowtie} Commande \right)$$

2.3 Requêtes sous SQL

Liste des ordinateurs portables en dessous de 1000 euros

```
   SELECT Nom,prix,NomC
FROM Appartient,Produit,Categorie
WHERE IDProduit=IDprod and IDCategorie=IDcat
  and prix<1000 and NomC like "PC portable"
```

Requête utilisant l'opérateur IN : Liste des commandes en Europe

```
   SELECT Produit.Nom,CompteClient.Nom,nCommande,DateCommande,Pays
FROM Adresse,Habite,
(Produit join Commande on produit.IDProduit=Commande.IdProduit)join CompteClient on Com-
mande.IdClient=CompteClient.IDClient
  WHERE CompteClient.IdClient=Habite.IDClient and Habite.IDAdresse=Adresse.IDAdresse
  and Pays in("Suède","France","Espagne","Allemagne","Italie")
```

Requête utilisant l'opérateur NOT IN : Liste des commandes hors Afrique

```
   SELECT Produit.Nom,CompteClient.Nom,nCommande,DateCommande,Pays
FROM Adresse,Habite,
  (Produit join Commande on produit.IDProduit=Commande.IdProduit)join CompteClient on Com-
mande.IdClient=CompteClient.IDClient
WHERE CompteClient.IdClient=Habite.IDClient and Habite.IDAdresse=Adresse.IDAdresse
  and Pays not in("Maroc","Sénégal")
```

Requête utilisant l'opérateur EXISTS : Les utilisateurs ayants effectué au moins une commande

```
   SELECT *
FROM CompteClient
WHERE EXISTS (SELECT *
                FROM Commande
                WHERE CompteClient.IDClient=Commande.IdClient )
```

Requête utilisant l'opérateur NOT EXISTS : faire l'inverse càd les client ayant effectué aucune commande

```
   SELECT *
FROM CompteClient
WHERE NOT EXISTS (SELECT *
                    FROM Commande
                    WHERE CompteClient.IDClient=Commande.IdClient )
```

Requêtes IMBRIQUES : Client ayant dépensé plus de 1000 euros

```
   SELECT *
FROM (SELECT sum(prix) as depense, CompteClient.Nom
      FROM (commande join CompteClient on Commande.IdClient=CompteClient.IDClient)join Pro-
duit on commande.IdProduit=Produit.IDProduit
      GROUP BY CompteClient.IDClient)
WHERE depense >=1000
```

Requête utilisant l'opérateur COUNT : Trouver les clients ayant effectués le plus de commandes (pour des offres promotionnelles)

SELECT **COUNT**(CompteClient.IDClient) as nb_commandes,CompteClient.Nom
FROM (Produit join Commande on produit.IDProduit=Commande.IdProduit)join CompteClient on Commande.IdClient=CompteClient.IDClient
GROUP BY (CompteClient.IdClient) ORDER BY nb_commandes DESC LIMIT 5

```
select count(CompteClient.IDClient) as nb_commandes,CompteClient.Nom
       from (Produit join Commande on produit.IDProduit=Commande.IdProduit)join CompteClient on Commande.IdClient=CompteClient.IDClient
           group by (CompteClient.IdClient) order by nb_commandes desc limit 5
```

	nb_commandes	Nom
1	9	Hugo
2	3	Lila
3	3	Yacout
4	3	Ghali
5	3	Nathalie

```
L'exécution s'est terminée sans erreur.
Résultat : 5 enregistrements ramenés en 11ms
À la ligne 1 :
select count(CompteClient.IDClient) as nb_commandes,CompteClient.Nom
                    from (Produit join Commande on produit.IDProduit=Commande.IdProduit)join CompteClient on Commande.IdClient=CompteClient.IDClient
                              group by (CompteClient.IdClient) order by nb_commandes desc limit 5
```

FIGURE 2.1 – Les clients ayant effectués le plus de commandes

JOINTURE AVEC 2 FOIS LA même TABLE : Liste des produits commandés par au moins deux utilisateurs Liste des utilisateurs avec au moins deux adresses

SELECT DISTINCT Produit.IdProduit,Produit.Nom
FROM **Commande x**,**Commande y**,Produit
WHERE x.IdClient !=y.IdClient and x.IdProduit=y.IdProduit and x.IdProduit=Produit.IDProduit

Requête utilisant l'opérateur GROUP BY : Les fournisseurs ayant fait le meilleure chiffre d'affaire

SELECT max(chiffre_affaire),*
FROM (SELECT Fournisseur.nomEntrep,sum(Prix) as chiffre_affaire,count(Produit.IdProduit) as nb_produits
FROM Fournit,Fournisseur,(Produit join commande on Produit.IDProduit=Commande.IdProduit)
WHERE Fournisseur.IDFournisseur=Fournit.IDFourn and Fournit.IDPro=Produit.IDProduit
GROUP BY Fournisseur.IDFournisseur)

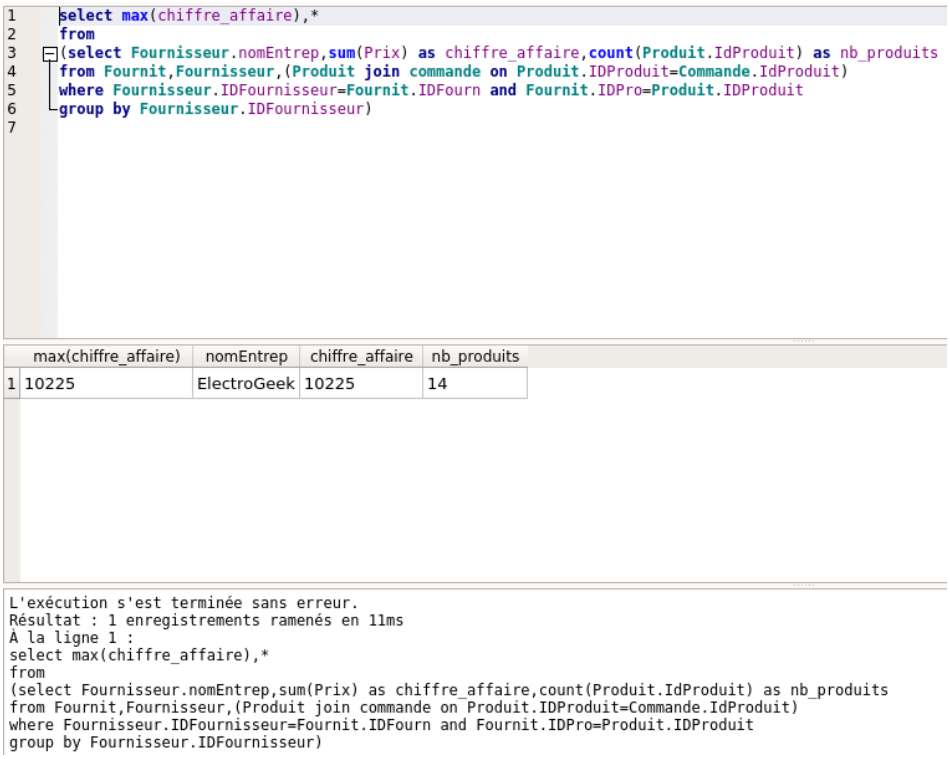

FIGURE 2.2 – Les fournisseurs ayant fait le meilleure chiffre d'affaire

SOUS REQUÊTES CORRELES : rechercher tous les fournisseur dont les ventes sont supérieur à 10 % des ventes totales.

SELECT *,0.1*SCA as "10 % du total des ventes"
FROM (SELECT Fournisseur.nomEntrep,sum(Prix) as chiffre_affaire,count(Produit.IdProduit) as nb_produits
FROM Fournit,Fournisseur,(Produit join commande on Produit.IDProduit=Commande.IdProduit)
WHERE Fournisseur.IDFournisseur=Fournit.IDFourn and Fournit.IDPro=Produit.IDProduit
GROUP BY Fournisseur.IDFournisseur)
,(SELECT sum(prix) as SCA
FROM (commande join Produit on Commande.IdProduit=Produit.IDProduit))
WHERE 0.1*SCA< chiffre_affaire

```
1   select *,0.1*SCA as "10 % du total des ventes"
2   from
3  (select Fournisseur.nomEntrep,sum(Prix) as chiffre_affaire,count(Produit.IdProduit) as nb_produits
4   from Fournit,Fournisseur,(Produit join commande on Produit.IDProduit=Commande.IdProduit)
5   where Fournisseur.IDFournisseur=Fournit.IDFourn and Fournit.IDPro=Produit.IDProduit
6   group by Fournisseur.IDFournisseur)
7  ,(SELECT sum(prix) as SCA
8   from (commande join Produit on Commande.IdProduit=Produit.IDProduit))
9   where 0.1*SCA<  chiffre_affaire
10
```

	nomEntrep	chiffre_affaire	nb_produits	SCA	10 % du total des ventes
1	Decathlon	4998	2	19325	1932.5
2	HyperX	2200	4	19325	1932.5
3	ElectroGeek	10225	14	19325	1932.5

```
L'exécution s'est terminée sans erreur.
Résultat : 3 enregistrements ramenés en 11ms
À la ligne 1 :
select *,0.1*SCA as "10 % du total des ventes"
from
(select Fournisseur.nomEntrep,sum(Prix) as chiffre_affaire,count(Produit.IdProduit) as nb_produits
from Fournit,Fournisseur,(Produit join commande on Produit.IDProduit=Commande.IdProduit)
where Fournisseur.IDFournisseur=Fournit.IDFourn and Fournit.IDPro=Produit.IDProduit
group by Fournisseur.IDFournisseur)
,(SELECT sum(prix) as SCA
from (commande join Produit on Commande.IdProduit=Produit.IDProduit))
where 0.1*SCA<  chiffre_affaire
```

FIGURE 2.3 – Tous les fournisseur dont les ventes sont supérieur à 10 pour cent des ventes totales.

Requête utilisant l'opérateur DIVISION : Liste des clients ayant commandé toutes les catégories

SELECT DISTINCT*
FROM CompteClient
WHERE **not EXISTS**(
SELECT *
FROM Categorie
WHERE **not EXISTS** (
SELECT *
FROM Commande,(Produit join Appartient on Produit.IDProduit=Appartient.IDprod)
WHERE Commande.IdProduit=Produit.IDProduit
and Categorie.IDCategorie=Appartient.IDcat
and CompteClient.IDClient=Commande.IdClient))

```
1   select DISTINCT* from CompteClient
2   where not EXISTS(
3   SELECT *
4   FROM Categorie
5   where not EXISTS (
6   select * from Commande,(Produit join Appartient on Produit.IDProduit=Appartient.IDprod)
7       where  Commande.IdProduit=Produit.IDProduit and Categorie.IDCategorie=Appartient.IDcat
8           and CompteClient.IDClient=Commande.IdClient
9   )
10  )
```

	IDClient	Nom
1	10604	Hugo

```
L'exécution s'est terminée sans erreur.
Résultat : 1 enregistrements ramenés en 8ms
À la ligne 1 :
select DISTINCT* from CompteClient
where not EXISTS(
SELECT *
FROM Categorie
where not EXISTS (
select * from Commande,(Produit join Appartient on Produit.IDProduit=Appartient.IDprod)
          where  Commande.IdProduit=Produit.IDProduit and Categorie.IDCategorie=Appartient.IDcat
                    and CompteClient.IDClient=Commande.IdClient
)
)
```

FIGURE 2.4 – Liste des clients ayant commandé toutes les catégories

Liste des fournisseurs fournissant toutes les catégories

```
    SELECT Fournisseur.nomEntrep
FROM Fournisseur
WHERE NOT EXISTS (
        SELECT *
        FROM Categorie
        WHERE NOT EXISTS (
            SELECT *
            FROM Fournit,Produit,Appartient
            WHERE Fournisseur.IDFournisseur=Fournit.IDFourn
              and Fournit.IDPro=Produit.IDProduit
              and Produit.IDProduit=Appartient.IDprod
              and Categorie.IDCategorie=Appartient.IDcat))
```

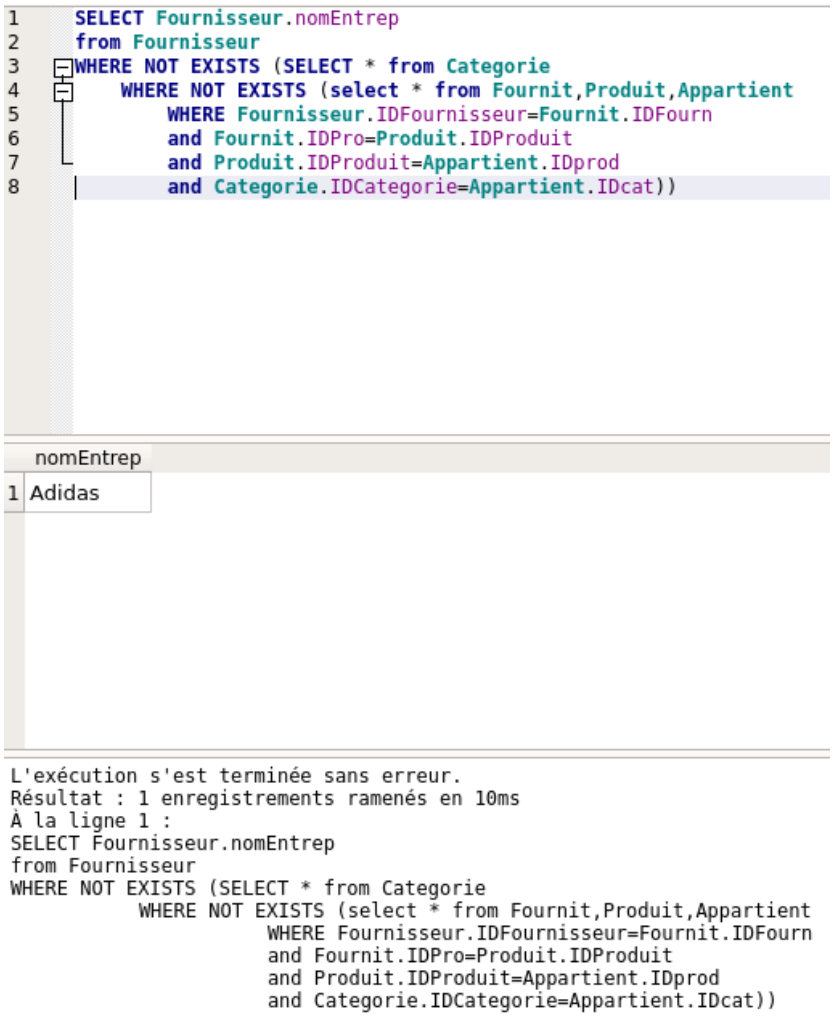

FIGURE 2.5 – Liste des fournisseurs fournissant toutes les catégories

3. Conclusion

Grâce à ce projet, nous avons pu découvrir et appliquer les différentes étapes nécessaires à la conception et l'implantation d'une base de données. Après s'être accordé sur un sujet qui nous intéressait, nous avons constitué notre modèle entité association, puis nous sommes passés à la création d'une base de données et enfin nous avons essayé d'imaginer plusieurs requêtes plausibles en les traduisant en algèbre relationnel dans un premier temps, puis en les implémentant sous SQL pour découvrir les résultats.